Key Design Features

- Synthesizable, technology independent VHDL Core
- 32-bit floating-point input
- Signed fixed-point or integer output
- Configurable word and fraction width up to 32 integer bits and 23 fraction bits
- IEEE 754 compliant
- High-speed fully pipelined architecture
- 2 clock-cycle latency

Applications

- Floating-point pipelines and arithmetic units
- Floating-point processors
- Interfacing between floating-point and fixed-point number systems

Pin-out Description

Pin name	*I/O*	*Description*	*Active state*
clk	in	Synchronous clock	rising edge
en	in	Clock enable	high
ieee_in [31:0]	in	Floating-point input in IEEE 754 format	data
fixed_out [dw - 1:0]	out	Signed fixed-point or integer output in [dw fw] format	data
ofl	out	Overflow flag	high
inf	out	Infinity flag	high
den	out	Denormalized number flag	high
nan	out	NaN flag	high

Generic Parameters

Generic name	*Description*	*Type*	*Valid range*
dw	Fixed-point word width	integer	2 ≤ dw ≤ 32
fw	Fixed-point fraction width	integer	0 ≤ fw ≤ 23 (fw < dw)

Block Diagram

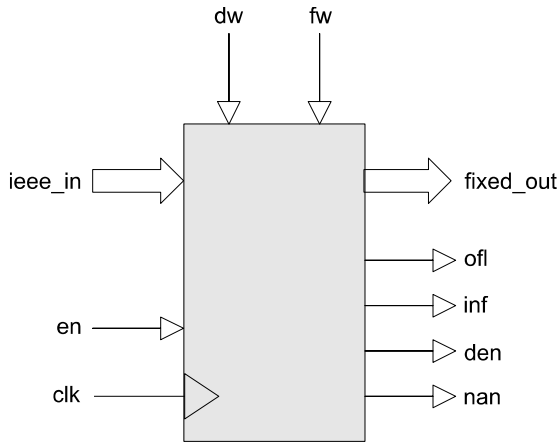
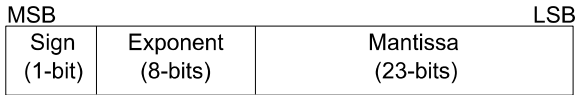


Figure 1: 32-bit Floating-point to Fixed-point converter

General Description

IEEE_TO_FIXED (Figure 1) is a high-speed fully pipelined conversion unit that accepts a 32-bit bit floating-point number as input and generates a fixed-point representation at the output. The input number is based on the IEEE 754 standard with the bits arranged in the following format:



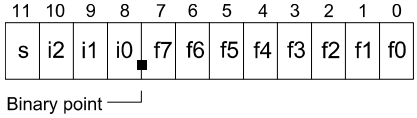
The real number representation of the floating-point number may be calculated as:

$$Value = -1(S) * 2^{(E-127)} * 1.\mathrm{M}$$

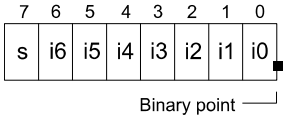
The fixed-point format is configured using the generic parameters *dw* and *fw*. The value *dw* specifies the width of the output word and *fw* specifies the number of fraction bits.

The output may be specified as either a signed fixed-point number or a signed integer. If a signed integer is preferred, then *fw* must be set to 0. In all cases *dw* must be at least 2 bits and *fw* must be less than *dw*.

As an example, to generate a 12-bit fixed-point output with 8 fraction bits the generic parameters must be set to: dw = 12, fw = 8. In this example the output word would be arranged as follows:



Alternatively, consider the case where the output format is an 8-bit signed integer. Setting dw = 8 and fw = 0, the output word would be:



In addition, there are some special cases that need to be observed. In the case of a positive floating-point input being larger than the maximum representable fixed-point number then the output will saturate to the maximum positive fixed-point number. Likewise, the output will saturate to the maximum allowable negative number if the a negative floating-point input is too large. In both instances of positive and negative overflow, the signal *ofl* will be asserted with the output value.

If the input floating-point value is a denormalized number, then the fixed-point output will be 0 and the *den* flag will be asserted high. If the input is a NaN then the fixed-point output will be set to 0 and the *nan* flag asserted. Finally, the case of the input being positive or negative Infinity is treated in the same manner as a positive or negative overflow, with additional *inf* flag also being asserted.

All values are sampled on the rising clock-edge of *clk* when *en* is high. The function has a 2 clock-cycle latency.

Functional Timing

Figure 2 demonstrates the conversion of two floating-point numbers. The first number is 0xC0FE6666 and the second is 0x40F80000 which represent the real numbers -7.95 and 7.75 respectively. The generic parameters have been set to: dw = 12, fw = 8. The results are available two clock cycles later.

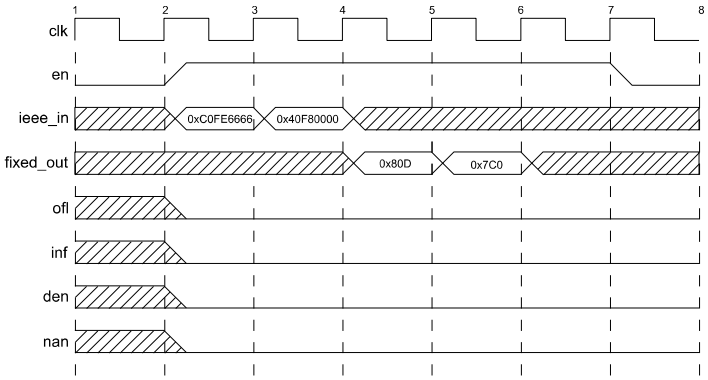


Figure 2: Floating-point to [12 8] fixed-point conversion

Figure 3 shows a sequence of floating-point exceptions that cause the various flags to be exercised. The first value of 0xC101999A represents the number -8.1. The largest negative number allowed in [12 8] format is -8 so this causes the overflow flag to be asserted. The next three values represent a denormalized number, +Infinity and NaN.

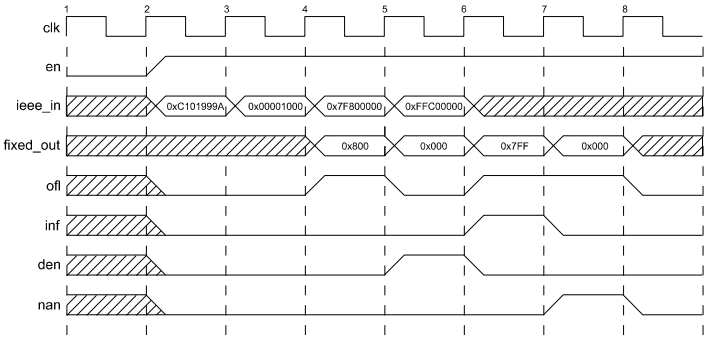


Figure 3: Floating-point to [12 8] fixed-point with flags

Source File Description

All source files are provided as text files coded in VHDL. The following table gives a brief description of each file.

Source file	*Description*
ieee_to_fixed.vhd	Top-level component
ieee_to_fixed_bench.vhd	Top-level test bench

Functional Testing

An example VHDL testbench is provided for use in a suitable VHDL simulator. The compilation order of the source code is as follows:

1. ieee_to_fixed.vhd
2. ieee_to_fixed_bench.vhd

The VHDL testbench instantiates the top-level component and the user may modify the generic parameters *dw* and *fw* as required

The simulation must be run for at least 2 ms during which time the 'ieee_to_fixed' component will receive an input stimulus of randomized floating-point numbers.

The simulation generates two text files called: *ieee_to_fixed_in.txt* and *ieee_to_fixed_out.txt*. These files respectively contain the input and output values captured during the test.

Synthesis

The source file 'ieee_to_fixed.vhd' is the only file required for synthesis. There are no sub-modules in the design.

The VHDL core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx Virtex 5 and the Altera Stratix III series of FPGA devices. The lowest and highest speed grade devices have been chosen in both cases for comparison. Trial synthesis results are shown with the generic parameters set to: dw = 32, fw = 23.

Resource usage is specified after Place and Route.

Copyright © 2011 www.zipcores.com

VIRTEX 5

Resource type	***Quantity used***
Slice register	78
Slice LUT	267
Block RAM	0
DSP48	0
Clock frequency (worst case)	250 MHz
Clock frequency (best case)	300 MHz

STRATIX III

Resource type	***Quantity used***
Register	78
ALUT	373
Block Memory bit	0
DSP block 18	0
Clock frequency (worse case)	250 MHz
Clock frequency (best case)	360 MHz

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision	30/04/2008
1.1	Updated synthesis results in line with minor source code changes	21/09/2011

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Development Software ***category:***

Click to view products by Zipcores ***manufacturer:***

Other Similar products are found below :

SRP004001-01 SW163052 SYSWINEV21 WS01NCTF1E W128E13 SW89CN0-ZCC IP-UART-16550 MPROG-PRO535E AFLCF-08-LX-CE060-R21 WS02-CFSC1-EV3-UP SYSMAC-STUDIO-EIPCPLR 1120270005 SW006021-2H ATATMELSTUDIO 2400573 2702579 2988609 SW006022-DGL 2400303 88970111 DG-ACC-NET-CD 55195101-101 55195101-102 SW1A-W1C MDK-ARM SW006021-2NH B10443 SW006021-1H SW006021-2 SW006022-2 SW006023-2 SW007023 MIKROE-730 MIKROE-2401 MIKROE-499 MIKROE-722 MIKROE-724 MIKROE-726 MIKROE-728 MIKROE-732 MIKROE-734 MIKROE-736 MIKROE-738 MIKROE-744 MIKROE-928 MIKROE-936 1120270002 1120270003 1120275015 NT-ZJCAT1-EV4